

pgGrid: uma Implementação de Fragmentação de Dados para o PostgreSQL

Gustavo A. Tonini, Frank Siqueira

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina
Florianópolis – SC – Brasil

`gustavotonini@gmail.com, frank@inf.ufsc.br`

Abstract. *As an organization grows, it needs to store great amounts of data and organize them in a way that allows its recovery. The goal of pgGrid is to offer an extension to PostgreSQL database management system (DBMS) that allows data fragmentation and distribution in the most convenient form among multiple database servers. It was necessary to modify the "pgcluster" tool to manage data location and to optimize distributed queries. Moreover, an extension to the data definition language (DDL) was proposed to define data distribution parameters and distributed system's topology.*

Resumo. *À medida que as organizações crescem, também cresce a necessidade de armazenar grandes massas de dados e organizá-los de uma forma que favoreça sua recuperação. A proposta deste trabalho é oferecer uma extensão ao sistema gerenciador de banco de dados (SGBD) PostgreSQL que permita a fragmentação e a distribuição dos dados da forma mais conveniente em vários servidores de banco de dados. Para isso foi necessário modificar a ferramenta "pgcluster" a fim de gerenciar a localização dos dados no sistema distribuído e otimizar as consultas. Além disso, foi proposta uma extensão à linguagem de definição de dados (DDL) para a definição dos parâmetros da distribuição dos dados e dos "sítios" que formam o sistema de banco de dados distribuído (SBDD).*

1. Introdução

Nos primórdios da computação, todo processamento era realizado de forma centralizada. Com o passar dos anos foi-se notando o potencial que as características dos sistemas distribuídos proporcionam. Sistemas de banco de dados distribuídos são sistemas distribuídos que armazenam, manipulam e organizam dados.

De acordo com [Ozsu 1999], as principais promessas e características dos sistemas de banco de dados distribuídos são:

- Administração transparente dos dados fragmentados nas máquinas que compõem o sistema;
- Melhoria no desempenho das consultas, através da paralelização das mesmas e de outras estratégias de otimização; e
- Fácil expansão e boa escalabilidade do sistema.

Muitos gerenciadores de banco de dados relacionais proprietários possuem funções relativas a sistemas deste tipo, podendo-se destacar: *Oracle Database*, *Microsoft SQL*

Server e *IBM DB2* [Bedoya 2001]. No entanto, existem poucas implementações disponíveis para uso no âmbito do software livre.

O presente artigo apresenta a implementação de uma extensão para o SGBD PostgreSQL que adiciona suporte para fragmentação e replicação de dados.

O trabalho está dividido em seis seções. As seções dois e três apresentam o pgGrid e detalham seu funcionamento. A quarta seção faz um comparativo com outras implementações disponíveis no mercado e as duas seções finais apresentam um caso de uso e os testes de desempenho realizados sobre o mesmo.

2. PostgreSQL e pgCluster

PostgreSQL é um sistema gerenciador de banco de dados de código-fonte aberto, multiplataforma, com mais de 15 anos de desenvolvimento. Segue o padrão SQL99 e seu gerenciador de transações garante as propriedades atomicidade, consistência, isolamento e durabilidade (ACID) [PostgreSQL 2009].

Dentre uma gama enorme de SGBDs relacionais e objeto-relacionais disponíveis no mercado, o PostgreSQL se sobressai devido à sua aceitação pela comunidade de software livre mundial, estabilidade e aderência ao padrão SQL.

O *pgCluster* é uma extensão do PostgreSQL que fornece replicação síncrona dos dados. A ferramenta possui duas funções principais:

- **Balanceamento de carga:** Um módulo recebe os comandos de consulta e transações e os distribui da forma mais conveniente, levando em consideração a carga (capacidade de processamento comprometida) dos servidores.
- **Alta disponibilidade:** É garantida através da replicação total dos dados; assim, se um servidor falha, os demais podem responder em seu lugar, desde que pelo menos um servidor de replicação continue no ar.

Conforme se pode facilmente perceber, o maior problema do pgCluster é a falta das funcionalidades de fragmentação, que exigem que o usuário replique totalmente as bases de dados para que o sistema funcione, degradando o desempenho e também desperdiçando recursos do sistema.

3. Extensões propostas ao pgCluster

Fragmentar os dados é uma necessidade básica na maioria dos ambientes distribuídos [Ozsu 1999]. Com este propósito, o pgGrid adiciona funções de fragmentação para o pgCluster e fornece comandos, como uma extensão à DDL, para definição dos fragmentos e alocação dos mesmos nos servidores.

Três grupos de instruções especiais foram adicionados à gramática:

- Adição e exclusão de servidores no sistema (CREATE SERVER);
- Definição de fragmentos (CREATE FRAGMENT);
- Alocação de fragmentos nos servidores (PLACE);

A Figura 1 apresenta alguns exemplos de uso dos comandos adicionados.

```
CREATE SERVER "site1" HOST "inf.ufsc.br" PORT 5432 RECOVERY PORT 7000;
CREATE FRAGMENT cliente_pessoa_fisica ON CLIENTE (cod_cliente, nm_cliente) where
tipo_pessoa = 'F';
```

```
CREATE FRAGMENT cliente_filial1 ON CLIENTE where cod_filial = 1;  
PLACE cliente_pessoa_fisica ON sitel;
```

Figura 1. Exemplo de uso dos comandos de definição do cluster

As informações definidas nos comandos supracitados ficam armazenadas nos catálogos do sistema. Quatro catálogos foram adicionados com este propósito: *pg_grid_site*, *pg_grid_fragment*, *pg_grid_fragment_attribute* e *pg_grid_allocation*;

Depois que os parâmetros foram definidos, o pgGrid mantém os servidores sincronizados, garantindo as regras de fragmentação explicitadas pelo administrador.

Sites e fragmentos podem ser definidos e alocados a qualquer momento, sem a necessidade de interrupção do funcionamento do sistema. Tal característica provê a escalabilidade necessária para aplicações reais.

As próximas seções explicam brevemente como o pgGrid trata os comandos de manipulação de dados, tornando a fragmentação possível.

3.1. Processamento dos comandos de manipulação de dados

Toda vez que um comando INSERT, UPDATE ou DELETE é submetido ao servidor, o mesmo é analisado e enviado para um módulo denominado *executor*. O executor foi alterado para verificar nos novos catálogos a necessidade de replicação da solicitação.

Quando o sistema identifica que um comando deve alterar dados de um ou mais sítios remotos, é iniciada uma transação distribuída onde o comando em questão é replicado para todos os sites envolvidos. A execução do comando somente é dada como concluída quando todos os sites confirmarem o sucesso da operação para o site que recebeu a solicitação. Se algum servidor reportar problemas na execução do comando, são enviadas mensagens para desfazer as alterações em todos os nodos envolvidos, segundo o protocolo de recuperação adotado pelo *pgCluster*.

3.2. Processamento das consultas distribuídas

Toda vez que um comando SELECT é submetido ao servidor, o mesmo é analisado, passa por otimizações para melhorar seu desempenho e depois é enviado para o mesmo módulo executor dos demais comandos. O executor das consultas foi alterado para consultar os catálogos do pgGrid e verificar a necessidade de solicitação de dados para outros sites do cluster.

Quando o sistema identifica que uma consulta necessita de dados externos (através de reduções [Ozsu 1999]), uma solicitação é enviada para cada site que contém dados usados pela consulta. Cada site processa as operações possíveis no seu conjunto de dados (provendo paralelização) e envia o resultado para o servidor que recebeu a solicitação, este é responsável por “juntar” os dados (através de operações de união e junção) e apresentar o resultado final ao usuário.

4. Outros SGBDs Distribuídos

Como parte do trabalho, foi realizada a avaliação de várias implementações de banco de dados distribuídos disponíveis no mercado.

Os gerenciadores avaliados foram: MySQL [MySql AB 2007], Oracle Database [Gopalakrishnan 2006] e IBM DB2 [Bedoya 2001]. A avaliação foi realizada segundo

alguns critérios relevantes para a manutenção de um sistema de banco de dados distribuído, de uma forma comparativa com o pgGrid, tais como sincronismo (as operações devem, obrigatoriamente, ser realizadas em todos os sites logo após à solicitação) e assincronismo (as operações podem ser realizadas em outro momento em alguns dos servidores).

A Tabela 1 lista os critérios escolhidos e a avaliação de cada produto quanto à compatibilidade com o mesmo.

Tabela 1. Comparativo entre as tecnologias de BDD

Característica	MySQL	Oracle	IBM DB2	pgGrid
Consultas distribuídas	Sim	Sim	Sim	Sim
Tecnologia multi-mestre	Sim	Sim	Sim	Sim
Replicação parcial	Não	Sim	Sim	Sim
Replicação síncrona	Sim	Sim	Sim	Sim
Replicação assíncrona	Não	Sim	Não	Não
Fragmentação	Sim	Sim	Sim	Sim
Integridade referencial entre tabelas armazenadas em servidores distintos	Não	Não	Não	Sim
Protocolo de commit distribuído	2PC	2PC	2PC	2PC

5. Caso de uso

Para demonstrar as funcionalidades do pgGrid, um esquema de BDD foi montado. O esquema armazena dados sobre produtos produzidos no estado de Santa Catarina, conforme ilustrado no modelo lógico da Figura 2.

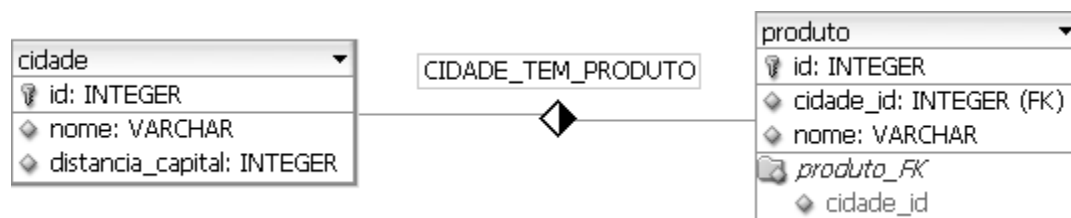


Figura 2. Modelo lógico correspondente ao caso de uso

Cinco sites pertencem ao sistema, localizados nas principais cidades do estado: Florianópolis (FLN), Joinville (JVL), Blumenau (BLU), Criciúma (CRI) e Chapecó (XAP). As regras de armazenamento são simples: cada cidade armazena seu cadastro e os produtos produzidos em sua região. A capital (FLN), além dos dados de seus produtos, armazena o cadastro de todas as cidades do estado para facilitar a listagem da tabela neste local. Os produtos possuem como atributos: código, nome e cidade de origem. As cidades possuem código, nome e a distância (em quilômetros) até a capital.

Depois que os cinco servidores foram configurados e colocados no ar, o modelo de dados e sua fragmentação foram definidos através dos comandos apresentados na Figura 3.

```

/*relação cidade*/
sc=# CREATE TABLE CIDADE (ID INT, NOME VARCHAR, DISTANCIA_CAPITAL INT);
sc=# ALTER TABLE CIDADE ADD CONSTRAINT PK_CIDADE PRIMARY KEY (ID);
/*relação produto*/
sc=# CREATE TABLE PRODUTO (ID INT, NOME VARCHAR, ID_CIDADE_ORIGEM INT);
sc=# ALTER TABLE PRODUTO ADD CONSTRAINT PK_PRODUTO PRIMARY KEY (ID);
  
```

```

sc=# ALTER TABLE PRODUTO ADD CONSTRAINT FK_PRODUTO_CIDADE FOREIGN
sc=# KEY (ID_CIDADE_ORIGEM) REFERENCES CIDADE (ID);
/*definição e alocação dos fragmentos dos produtos*/
sc=# CREATE FRAGMENT PRODUTO_FLN ON PRODUTO WHERE ID_CIDADE_ORIGEM=1;
sc=# PLACE PRODUTO_FLN ON FLN;
sc=# CREATE FRAGMENT PRODUTO_JVL ON PRODUTO WHERE ID_CIDADE_ORIGEM=2;
sc=# PLACE PRODUTO_JVL ON JVL;
sc=# CREATE FRAGMENT PRODUTO_BLU ON PRODUTO WHERE ID_CIDADE_ORIGEM=3;
sc=# PLACE PRODUTO_BLU ON BLU;
sc=# CREATE FRAGMENT PRODUTO_CRI ON PRODUTO WHERE ID_CIDADE_ORIGEM=4;
sc=# PLACE PRODUTO_CRI ON CRI;
sc=# CREATE FRAGMENT PRODUTO_XAP ON PRODUTO WHERE ID_CIDADE_ORIGEM=5;
sc=# PLACE PRODUTO_XAP ON XAP;
/*definição e alocação dos fragmentos das cidades*/
CREATE FRAGMENT CIDADE_FLN ON CIDADE;
PLACE CIDADE_FLN ON FLN;
CREATE FRAGMENT CIDADE_JVL ON CIDADE WHERE ID=2;
PLACE CIDADE_JVL ON JVL;
CREATE FRAGMENT CIDADE_BLU ON CIDADE WHERE ID=3;
PLACE CIDADE_BLU ON BLU;
CREATE FRAGMENT CIDADE_CRI ON CIDADE WHERE ID=4;
PLACE CIDADE_CRI ON CRI;
CREATE FRAGMENT CIDADE_XAP ON CIDADE WHERE ID=5;
PLACE CIDADE_XAP ON XAP;

```

Figura 3. Comandos utilizados na definição do caso de uso do pgGrid

Foram realizados testes inserindo as cidades e vários produtos no sistema. Depois todos os sites foram reiniciados no modo centralizado (sem sincronia com o cluster). Cada site continha somente os dados da cidade que representava, atestando o funcionamento do pgGrid.

6. Testes de desempenho

Os testes executados no caso de uso abordaram principalmente as garantias da integridade e reconstrução dos dados distribuídos no sistema. Também foram realizados testes para examinar o desempenho das operações dentro do sistema distribuído utilizado no caso de uso, fazendo uma comparação com o desempenho do pgCluster com a mesma configuração e massa de dados.

Buscando resultados mais precisos, foi gerada uma massa de dados com 500000 produtos que foi dividida em partes iguais nos sites do sistema, de acordo com as regras de fragmentação definidas anteriormente. Os testes foram realizados utilizando dois servidores (três sites alocados no servidor A e dois no servidor B) conectados a um switch de 100 Mbps (Fast Ethernet).

As operações utilizadas nos testes estão listadas na Tabela 2, e correspondem ao tempo das operações, carga dos servidores, comunicação através da rede e dos acessos aos dispositivos de armazenamento.

Tabela 2. Resultados dos testes de desempenho

Parâmetro	pgCluster	pgGrid
Tempo para atualização (UPDATE) na relação CIDADE	1.8s	0.9s
SELECT * FROM PRODUTO	5.3s	16.5s
SELECT * FROM PRODUTO WHERE NOME = 'PROD500001'	4s	2.3s

Pacotes de rede transmitidos durante a atualização	16	4
Carga de trabalho durante a consulta	20%	38%
Espaço médio ocupado pelo banco de dados em cada site	30MB	7.57MB

Pode-se notar, através dos dados obtidos, que a fragmentação nos dados melhorou as operações de atualização, pois não é necessário acessar todos os sites do sistema quando um item de dado é modificado. No entanto, a maioria das operações de consulta envolvendo mais de um site teve seu desempenho degradado devido à necessidade de juntar os dados distribuídos entre os servidores. No caso do pgCluster, com os dados totalmente replicados, apenas uma consulta local resolve o problema.

A grande melhoria notada pela fragmentação se refere ao desempenho das atualizações e consultas distribuídas. Com a remoção das réplicas desnecessárias, os recursos utilizados em cada servidor podem ser configurados de uma forma mais otimizada para atender às necessidades dos seus usuários.

7. Conclusões

Com a crescente necessidade de compartilhamento de informações, há uma demanda muito grande para o armazenamento das mesmas. Na maioria das vezes, os dados ficam centralizados em grandes servidores. No entanto, com volumes muito grandes, a centralização torna o processo de recuperação e carga dos dados ineficiente [Ozsu 1999].

Para resolver tal problema, os sistemas distribuídos entram em jogo fornecendo soluções escaláveis e paralelizáveis. Este trabalho propõe uma solução deste tipo no contexto de banco de dados relacionais e objeto-relacionais.

A proposta apresentada no início do trabalho mostrou-se viável no decorrer do desenvolvimento do mesmo. Tal viabilidade foi ilustrada pelo caso de uso e pelos testes de desempenho. No entanto, o módulo planejador de consultas precisa ser melhorado para usufruir mais da paralelização que a fragmentação proporciona e o software deve passar por testes mais rigorosos antes de ser colocado em produção em casos reais.

Como próximo passo planeja-se implementar as otimizações de consultas que o ambiente distribuído possibilita, tais como a paralelização de subconsultas.

Repositório do projeto

PgGrid é um software de código-fonte aberto, disponível pela licença GPL, que pode ser baixado do site <http://pgfoundry.org/projects/pggrid/>.

Referências

- Ozsu, M. Tamer (1999) “Principles of distributed database systems”. New Jersey, United States, Prentice Hall, p. 7-10.
- PostgreSQL Development Group (2009). “About PostgreSQL”, <http://www.postgresql.org/>.
- Bedoya, H. et al (2001). “Advanced Functions and Administration on DB2 Universal Database for iSeries”, <http://www.redbooks.ibm.com/abstracts/sg244249.html>.
- MYSQL AB (2007). “MySQL 5.0 Reference Manual”, <http://dev.mysql.com/doc/refman/5.0/>.
- GOPALAKRISHNAN, K “Oracle Database 10g Real Application Clusters Handbook”. Columbus: McGraw-Hill Osborne Media (Oracle Press), 2006, ISBN 9780071465090.